Object-Oriented Programming I (CSE 121)
Week # 11
(Polymorphism in C++)

Objective(s)

In this lecture, you will learn:

1. What is Polymorphism?
2. Why and when to use Polymorphism?
3. Program examples of Polymorphism
4. Types of Polymorphism

Polymorphism in C++

Object-Oriented Programming (OOP) has four core concepts which are listed below:

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

In this lecture, we will explore all about Polymorphism in C++.

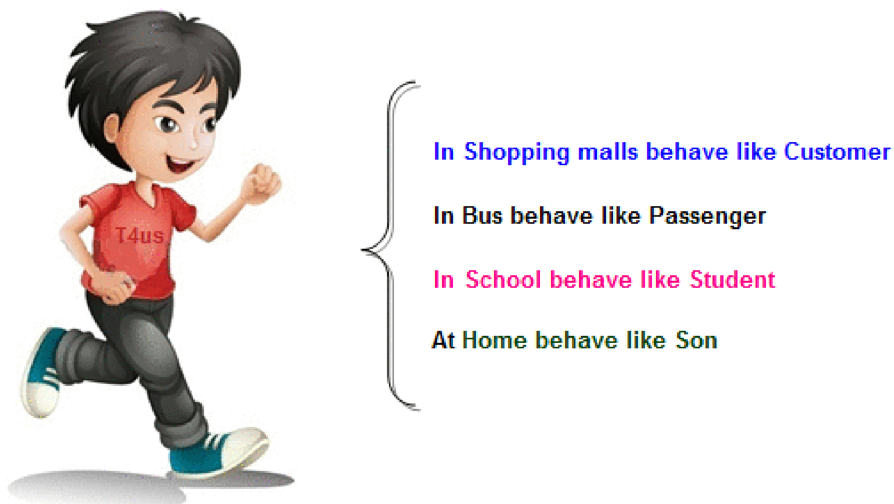
Polymorphism is one of the four pillars of object-oriented programming. Polymorphism means having many forms. It can be defined as the technique by which an object can take many forms depending on the situation.

Polymorphism is derived from 2 greek words: **poly** and morphs. The word "poly" means many and **morphs** means forms. So, polymorphism means many forms.

In programming terms, we can say that an object can behave differently in different conditions.

Real life example of Polymorphism in C++

Example 1: Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here, one person has different-different behaviors.



Example 2: A Person who knows more than two languages he can speak in a language which he knows. Here person is Object and speak is polymorphism.

Example 3: A woman can take many roles in different situations. To a child, she is a mother, a homemaker at home, a worker in the office, etc. So, a woman assumes different roles and exhibits different behavior in different conditions.

Type of polymorphism

- Compile time polymorphism
- Run time polymorphism

Compile time polymorphism

In C++ programming you can achieve compile time polymorphism in two ways, which is given below;

- Method overloading
- Method overriding

Method/Function Overloading in C++

A function is said to be overloaded when we have more than one function with the same name but different parameter types or a different number of arguments.

Thus, a function can be overloaded based on the parameter types, the order of parameters and the number of parameters.

Note: Two functions having the same name and same parameter list but different return type is not an overloaded function and will result in a compilation error if used in the program.

Example of Method/Function Overloading in C++

```
#include<iostream>
using namespace std;
class Addition{
public:
   void sum(int a, int b){
      cout << "a + b = " << a + b << endl;
   }
   void sum(double a, double b){
      cout << "a + b = " << a + b << endl;
   }
   void sum(int a, double b){
      cout << "a + b = " << a + b << endl;
   }
   void sum(double a, int b){
      cout << "a + b = " << a + b << endl;
   }
   void sum(int a, int b, int c){
      cout << "a + b + c = " << a + b << endl;
   }
   void sum(string a, string b){
      cout << "a + b = " << a + b << endl;
   }
};
```

```
int main()
{
   Addition obj;
   obj.sum(10,20);
   obj.sum(10.5,11.7);
   obj.sum(10,11.7);
   obj.sum(10.5,11);
   obj.sum(10,11,20);
   obj.sum("Hello ", "World");
   return 0;
}
```

In the above example, we overload the function **sum**() based on the type of parameters, sequence or order of parameters and number of parameters.

We have an operator “+” that is the binary addition operator which behaves differently when the operands change.

For Example, when both the operands are numeric, it performs addition. On the other hand, when the operands are string, it acts as concatenation operator.

Thus polymorphism, in a nutshell, means an entity taking up many forms or behaving differently under different conditions.

Benefits of Function Overloading

The main benefit of function overloading is that it promotes code reusability. We can have as many functions as possible with the same name as long as they are overloaded based on the argument type, argument sequence and the number of arguments.

By doing this it becomes easier to have different functions with the same name to represent the behavior of the same operation in different conditions.

If function overloading was not present, then we would have had to write too many different kinds of functions with different names, thus rendering the code unreadable and difficult to adapt.

Method Overriding in C++

Define any method in both **base class** and **derived class** with same name, same parameters or signature, this concept is known as **method overriding**. In below example same method **show()** is present in both **base** and **derived class** with same name and signature.

Example of Method/Function Overriding in C++

```
#include<iostream>
using namespace std;
class One
{
public:
   void show()
   {
      cout << "Hello from base class." << endl;
   }
};

class Two : public One
{
public:
   void show()
   {
      cout << "Hello from derived class." << endl;
   }
};
```

```
int main()
{
   One obj;
   obj.show();
   Two ob;
   ob.show();
   return 0;
}
```